

tb_cocotb.v

AUTHORS

JAY CONVERTINO

DATES

2024/12/10

INFORMATION

Brief

Test bench wrapper for cocotb

License MIT

Copyright 2024 Jay Convertino

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

tb_cocotb

```
module tb_cocotb #(
  parameter
    FIFO_DEPTH
    =
    256,
  parameter
    BYTE_WIDTH
    =
    1,
  parameter
    COUNT_WIDTH
    =
    8,
  parameter
```

```

FWFT
=
0,
parameter
RD_SYNC_DEPTH
=
0,
parameter
WR_SYNC_DEPTH
=
0,
parameter
DC_SYNC_DEPTH
=
0,
parameter
COUNT_DELAY
=
1,
parameter
COUNT_ENA
=
1,
parameter
DATA_ZERO
=
0,
parameter
ACK_ENA
=
1,
parameter
RAM_TYPE
=
"block"
) ( input rd_clk, input rd_rstn, input rd_en, output rd_valid, output [(BYT

```

Wrapper to interface with dut, FIFO

Parameters

FIFO_DEPTH parameter	Depth of the fifo, must be a power of two number(divisable aka $256 = 2^8$). Any non-power of two will be rounded up to the next closest.
BYTE_WIDTH parameter	How many bytes wide the data in/out will be.
COUNT_WIDTH parameter	Data count output width in bits. Should be the same power of two as fifo depth(256 for fifo depth... this should be 8).
FWFT parameter	1 for first word fall through mode. 0 for normal.
RD_SYNC_DEPTH parameter	Add in pipelining to read path. Defaults to 0.
WR_SYNC_DEPTH parameter	Add in pipelining to write path. Defaults to 0.
DC_SYNC_DEPTH parameter	Add in pipelining to data count path. Defaults to 0.
COUNT_DELAY parameter	Delay count by one clock cycle of the data count clock. Set this to 0 to disable (only disable if read/write/data_count are on the same clock domain!).
COUNT_ENA parameter	Enable the count output.
DATA_ZERO parameter	Zero out data output when enabled.

ACK_ENA <small>parameter</small>	Enable an ack when data is requested.
RAM_TYPE <small>parameter</small>	Set the RAM type of the fifo.

Ports

rd_clk	Clock for read data
rd_rstn	Negative edge reset for read.
rd_en	Active high enable of read interface.
rd_valid	Active high output that the data is valid.
rd_data	Output data
rd_empty	Active high output when read is empty.
wr_clk	Clock for write data
wr_rstn	Negative edge reset for write
wr_en	Active high enable of write interface.
wr_ack	Active high when enabled, that data write has been done.
wr_data	Input data
wr_full	Active high output that the FIFO is full.
data_count_clk	Clock for data count
data_count_rstn	Negative edge reset for data count.
data_count	Output that indicates the amount of data in the FIFO.

INSTANTIATED MODULES

dut

```
fifo #(
    FIFO_DEPTH(FIFO_DEPTH),
    BYTE_WIDTH(BYTE_WIDTH),
    COUNT_WIDTH(COUNT_WIDTH),
    FWFT(FWFT),
    RD_SYNC_DEPTH(RD_SYNC_DEPTH),
    WR_SYNC_DEPTH(WR_SYNC_DEPTH),
    DC_SYNC_DEPTH(DC_SYNC_DEPTH),
    COUNT_DELAY(COUNT_DELAY),
    COUNT_ENA(COUNT_ENA),
    DATA_ZERO(DATA_ZERO),
    ACK_ENA(ACK_ENA),
    RAM_TYPE(RAM_TYPE)
) dut ( .wr_clk(wr_clk), .wr_rstn(wr_rstn), .wr_en(wr_en), .wr_ack(wr_ack),
```

Device under test,fifo